Rendu 2 : Evaluation des requettes en étoiles - NOSQL

TRINQUART Matthieu et SANCHEZ Martin

Contexte

Lors des dernier rendu nous avions realiser un dictionaire et un index. Le dictionaire nous permet d'associer une chaine de caractère en un Integer. L'index consiste à faire des arbres de syntaxiques (SOP,SPO,PSO,POS,OPS,OSP) et on devra parcourir chaqu'un de ces arbres pour récupérer une valeur de Object,Predicat ou subject.

Le principe de ce rendu est lire un ensemble de requèt de récupèrer les object, subject ou le prédicat (dans le cadre du TP on demande de récupérer juste des subjects) et nous devons utilisé le principe de la requèt en étoiles pour retourner l'intersection entre 2 requèt.

Evaluation des requetes en étoiles

Nous avons une fonction getFromMap dans index qui prend en paramètre une hashmap (qui correspond à 1 des 6 arbres syntaxique de index) et qui parcour cette arbre en fonction des info1 et info2 qui sont soit des object, predicat ou subject

```
1 private ArrayList<Integer> getFromMap(HashMap<Integer, HashMap<Integer,
    ArrayList<Integer>>> map, String info1, String info2) {
2     ArrayList<Integer> ret = new ArrayList<>() ;
3     int i1 = Dictonnary.getInstance().encode(info1) ;
4     int i2 = Dictonnary.getInstance().encode(info2) ;
5     if(i1!=-1 && i2 != -1){
6         if (map.get(i1) != null) {
7             if (map.get(i1).get(i2)  != null) {
```

```
8                   ret.addAll(map.get(i1).get(i2)) ;
9               }
10          }
11      }
12      return ret;
13  }
```

Nous avons une fonction getFromPOS(String predicat, String object) qui prend en paramètre un object et un predicat qui va utiliser la fonction getFromMap pour parcrourir l'arbre syntaxique POS.

```
1  public ArrayList<Integer> getFromPOS(String predicat, String object) {
2      ArrayList<Integer> ret = new ArrayList<>(getFromMap(pos, predicat,
       object)) ;
3      return ret;
4  }
```

Cette fonction permet de récupérer les valeurs subject, predicat et object du StatementPattern et utilise la fonction getFromPOS de Index pour retourner le résultat de la requèt

```
1  private static ArrayList<Integer> RequestResult(StatementPattern sp) {
2      Value s = sp.getSubjectVar().getValue();
3      Value p = sp.getPredicateVar().getValue();
4      Value o = sp.getObjectVar().getValue();
5      return new ArrayList<>(Index.getInstance().getFromPOS(p.toString(), o.
       toString()));
6      }
```

Cette fonction prend en paramètre 2 ArrayList d'Integer qui représente à 2 reponses de requèt et fait l'intersection pour récupèrer l'intersection entre les deux resultat (ce qui correspond à la requèt en étoile)

```
1  private static ArrayList<Integer> intersect(ArrayList<Integer> a1,
       ArrayList<Integer> a2) {
2      ArrayList<Integer> ret = new ArrayList<>();
3      for(Integer i : a1) { if (a2.contains(i)) { ret.add(i) ; } }
4      return ret;
5  }
```

Pour la lecture des requettes nous avons utilisé la fonction processAQuery(ParsedQuery query) donné dans le main initial. On recupere une ParsedQuery, un objet qui représente une requette qu'on evalue grace à notre fonction RequestResult qui va retourner une ArrayList de String qui correspond au résultats de la requèt. On fait ensuite l'intersection entre le résultat courant (optenu par les précédente requèt) et le resultat de la request qui vient d'être exécuter.

Il retourne ensuite la résultat des requèt.

Nous affichons ensuite le résultat en décodant le résultat grace à la fonction de Dictionnary.

```
1     public static ArrayList<Integer> evaluateStarRequest(ParsedQuery query)
      {
2         ArrayList<Integer> results = new ArrayList<>() ;
3         List<StatementPattern> patterns = StatementPatternCollector.process
      (query.getTupleExpr());
4         for(StatementPattern sp : patterns) {
5             if (results.isEmpty()) { results.addAll(RequestResult(sp)) ;}
6             else { results = EvaluateRequest.intersect(results,
      RequestResult(sp)) ; }
7         }
8         return results ;
9     }
```

Comment pourrait on evaluer nimporte quelle requette

Nous avons pris le temps de reflechir a une facon de resoudre toutes les requettes en plus des requettes en etoile

Pour cela nous avons pense creer un objet 'Result' qui serai le resultat d'une requette.

```
1 public class Result {
2     private Integer s, p, o;
3     public Result(Integer s, Integer p, Integer o) {
4         this.s = s; this.p = p; this.o = o;
5     }
```

Pour cela nous avons fait une classe Result contenant le subject, predicat et object. Il suffira de regarder quelle de ces 3 valeurs est null et appeler getFromMap avec diférent arbre (SOP,PSO,etc..) en fonction de la valeur null de s, p, o.